1. **微机系统的定义**

微型计算机简称微机，俗称电脑，其准确的称谓应该是微型计算机系统。它可以简单地定义为：在微型计算机硬件系统的基础上配置必要的外部设备和软件构成的实体。

1. 通用台式机是一个最常见的微型计算机系统：

主板、电源、硬盘、内存、显卡、CPU、键盘鼠标、显示器、摄像头、操作系统、软件

CPU是整个系统的运算核心，CPU的运算速度决定了系统的性能。

1. 除了台式机，笔记本、平板、手机也是微机系统

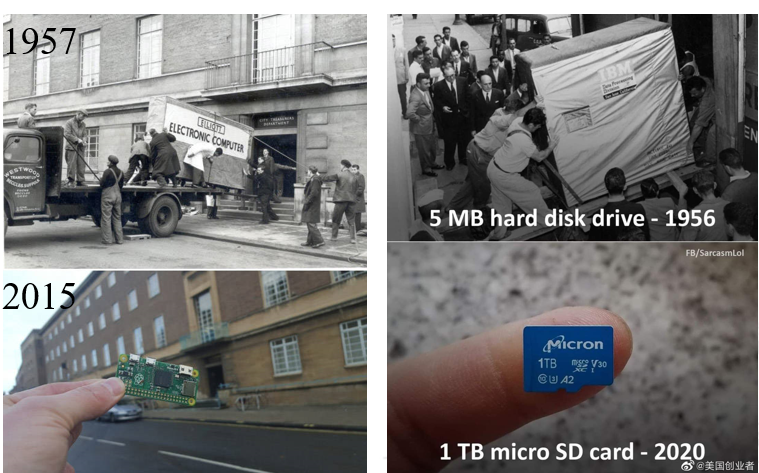
还有什么样的微机系统呢？微机系统会像什么方向发展呢？要回答这个问题，就需要了解微机系统的前生今世。

**计算机的发展：**

计算能力是人类文明发展和进步的重要能力

1. 初始的计算由机械的方法来完成，后又有了继电器计算机，统称为第零代计算机
2. 第一代（1946～1955年）电子管计算机
3. 第二代（1955～1965年）晶体管计算机
4. 第三代（1965～1980年）集成电路计算机
5. 第四代（1980年至今） 大规模集成电路计算机
6. 第五代 … 量子计算机、超导计算机、光计算机…

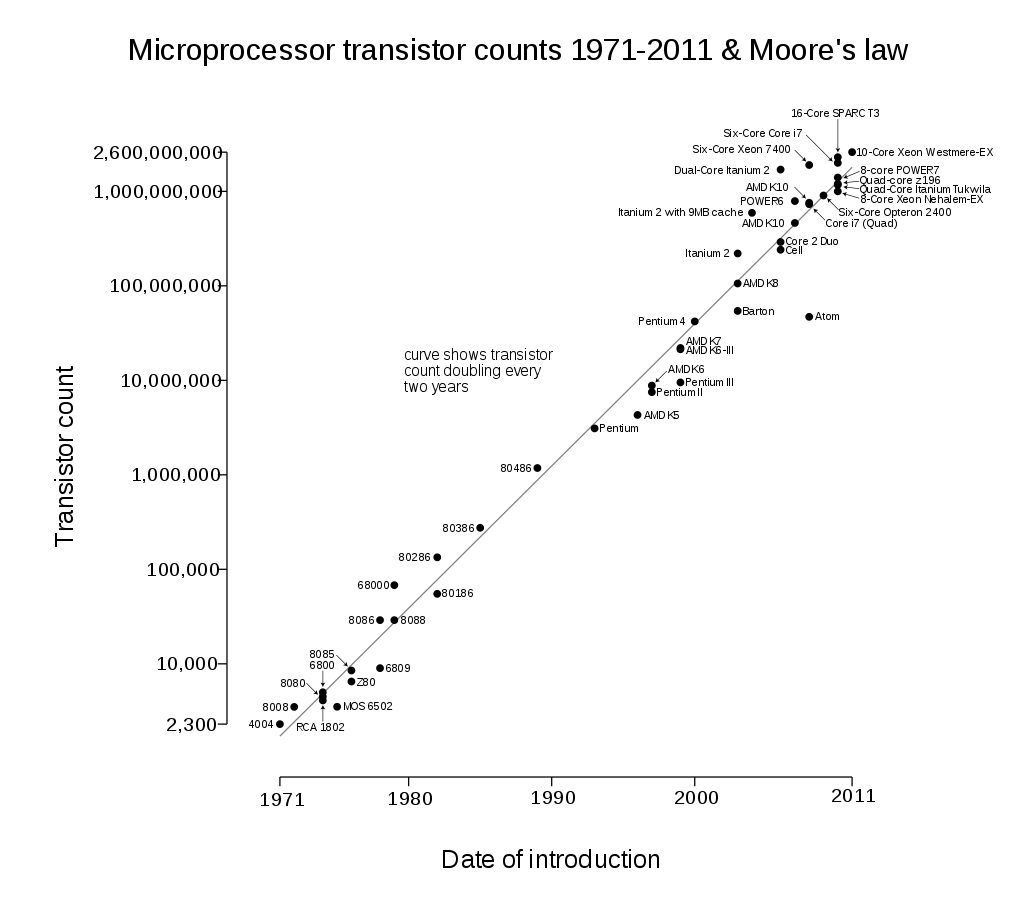
可以通过图片对比第二代晶体管计算机与当代大规模集成电路计算机的差别，直观的感受计算机技术的发展：



**摩尔定律**：

摩尔定律是由英特尔（Intel）创始人之一戈登·摩尔（Gordon Moore）提出来的，内容是：

当价格不变时，集成电路上可容纳的元器件的数目，约每隔18-24个月便会增加一倍，性能也将提升一倍。

从1971年到2011年之间CPU的发展，基本符合摩尔定律

摩尔定律是内行人摩尔的经验之谈，汉译名为“定律”，但并非自然科学定律，它一定程度揭示了信息技术进步的速度。

但是，随着新工艺节点的不断推出，晶体管中原子的数量已经越来越少，种种物理极限制约著其进一步发展。比如当闸极长度足够短的时候，量子隧穿效应就会发生，会导致漏电流增加。

随着计算机技术的发展，微机系统课分为通用微机和专用微机两类。

**通用微机**：PC、服务器、工作站等，用于办公或者计算机辅助设计等

**专用微机**：单片机、数字信号处理器、嵌入式系统等，应用在工业、交通、安防、家电、机器人、手机等各个领域。完成特定的功能。可靠性高、实时性强。程序相对简单、处理数据量小。

**嵌入式系统**是一种广泛应用的专用微机系统：

定义：

以应用为中心、计算机技术为基础，软、硬件可裁剪，适应应用系统对功能、可靠性、成本、体积、功耗严格要求的专用计算机系统。

嵌入式系统是一个硬件与软件的集合体：



硬件：

嵌入式处理器、外设等

软件：

操作系统软件、应用程序

嵌入式系统技术本质：

“内含计算机”、“嵌入到对象体系中”、“满足对象智能化控制要求”

嵌入系统在各个场所都得到了广泛的应用：

消费类电子产品、办公自动化产品、控制系统与工业自动化、 生物医学系统、现场仪器、网络通信设备、电信设备等。

**嵌入式微处理器**是嵌入式系统的核心：

**微处理器**（Embedded MPU；MPU = Microprocessor Unit）：MPU的基础是通用计算机中的CPU。与工控机相比，MPU具有体积小、重量轻、成本低及可靠性高的优点，但电路板上必须包括ROM、RAM、总线接口、各种外设等器件，从而降低了系统的可靠性，技术保密性也差。MPU及其存储器、总线、外设等安装在一块电路板上，称为单板计算机。

**微控制器**（MCU = Mircocontroller Unit）：MCU又称单片机，顾名思义，将整个计算机系统集成到一块芯片中。MCU一般以某种MPU为核心，芯片内部集成ROM/RAM/EEPROM/总线/总线逻辑/定时器/计数器/看门狗/脉宽调制输出/串行口/AD/DA等等各种必要功能和外设。与MPU相比，MCU最大的特点是单片化，体积大大减少，从而使功耗和成本下降，可靠性提高。MCU是目前嵌入式系统工业的主流。MCU的片上外设资源一般比较丰富，适合于控制，因此称为MCU。

**DSP处理器**（DSP = Digital Signal Processor）：DSP对系统结构和指令进行了特殊设计，使其适合于执行DSP算法，编译效率较高，指令执行速度也较高。在数字滤波、FFT、频谱分析等方面，DSP算法正大量进入嵌入式领域。

**片上系统**（SoC = System on Chip）：随着EDI的推广，VLSI设计的普及化及半导体工艺的迅速发展，在一个硅片上实现一个更为复杂的系统的时代已来临，这就是SoC。这样以来，除了个别无法集成的器件以外，整个嵌入式系统大部分都可集成到一块或几块芯片中，应用系统电路板将变得很简洁，对于减小体积和功耗，提高可靠性非常有利。

**外设**==外部设备/外围设备==Peripheral==硬件接口

外部：主要指的是除了嵌入式系统中主要的CPU之外的

设备：某种硬件功能模块

ADC、DAC、GPIO、UART、IIC、USB、SPI、PWM、TIMER等

**嵌入式软件**（固件）：

**前后台系统**：对于基于芯片开发来说，应用程序一般是一个无限的循环，可成为前后台系统或超循环系统。循环中调用相应的函数完成相应的操作，这部分可以看成后台行为；中断服务程序处理异步事件，这部分可以看成前台行为。后台也可以叫做任务级，前台也可以叫做中断级。时间相关性很强的关键操作一定是靠中断服务程序来保证的。

**实时操作系统（RTOS）：**在RTOS中，每个任务均有一个优先级，RTOS根据各个任务的优先级，动态地切换各个任务，保证实时性要求。实时多任务操作系统，以分时方式运行多个任务，看上去好像是多个任务“同时”运行。任务之间的切换应当以优先级为根据，只有优先服务方式的RTOS才是真正的实时操作系统，时间分片方式和协作方式的RTOS并不是真正的“实时”。

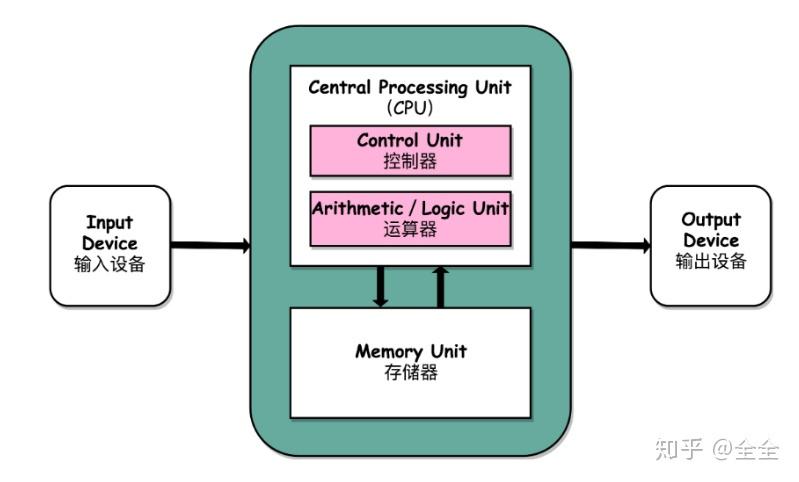
TM4C1294处理器简介：

基于ARM内核的处理器

带有各种外设：ADC、GPIO、UART、IIC、USB、SPI、PWM、TIMER

是一个典型的微控制器

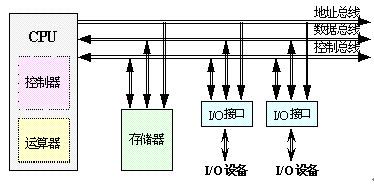
**冯·诺依曼体系结构:**



* 计算机处理的数据和指令用二进制数表示；
* 采用存储程序方式，指令和数据存储在存储器中；
* 顺序执行程序的每一条指令；
* 由存储器、运算器、控制器、输入设备和输出设备五大部件组成计算机系统，并规定了这五部分的基本功能。
* 存储器：用来存放数据和程序
* 运算器：主要运行算数运算和逻辑运算，并将中间结果暂存到运算器中
* 控制器：主要用来控制和指挥程序和数据的输入运行，以及处理运算结果
* 输入设备：用来将人们熟悉的信息形式转换为机器能够识别的信息形式，常见的有键盘，鼠标等
* 输出设备：可以将机器运算结果转换为人们熟悉的信息形式，如打印机输出，显示器输出等

其中，运算器和控制器合为处理器（CPU），输入输出设备合称为I/O设备

为了简化各个部件的相互连接，现代计算机广泛采用**总线**结构。总线具有组合灵活、扩展方便的特点。



**总线**：

物理上：传递信息的一组公用导线

功能上：传送信息的公共通道

使用限制：任一时刻，在总线上只能传递一种信息，只能有一个部件在发送信息，但可以有多个部件在接收信息。

**系统总线**是指微机系统中，微处理器与存储器、IO设备进行信息交换的公共通道

**地址总线AB**

输入/输出将要访问的内存单元或I/O端口的地址

地址线的多少决定了系统直接寻址存储器的范围

**数据总线DB**

CPU读操作时，外部数据通过数据总线送往CPU

CPU写操作时，CPU数据通过数据总线送往外部

数据线的多少决定了一次能够传送数据的位数

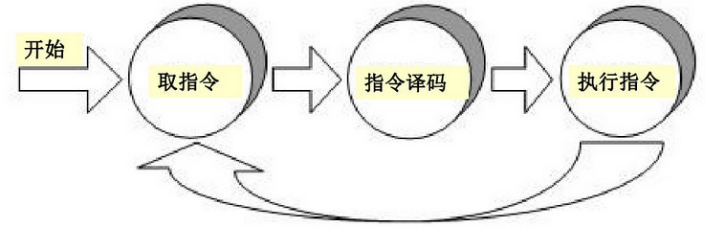
**控制总线CB**

协调系统中各部件的操作，有输出控制、输入状态等信号

控制总线决定了系统总线的特点，例如功能、适应性等

**CPU的工作原理：**

程序的执行过程实际上是不断地取出指令、分析指令、执行指令的过程。



（1）预先把指挥计算机如何进行操作的指令序列（就是程序）和原始数据输入到计算机内存中，每条指令中明确规定了计算机从哪个地址取数，进行什么操作，然后送到什么地方去等步骤。

（2）计算机在执行时，先从内存中取出第一条指令，通过控制器的译码器接收指令的要求，再从存储器中取出数据进行指定的运算和逻辑操作等，然后再按地址把结果送到内存中，如果需要向硬盘等存储设备存储数据，还需要将内存中的该数据存储到硬盘中。接下来取出第2条指令，在控制器的指挥下完成规定操作，依次进行下去，直到遇到停止指令。

（3）计算机中基本上有两股信息在流动，一种是数据，即各种原始数据、中间结果和程序等，另一种信息是控制信息，它控制机器的各种部件执行指令规定的各种操作。

举例：存储器中一条命令，假设这条命令执行这样一条动作，123+321，“+”表示要做的事情（相加），123和321是参数相加动作的两个数



（1）通过命令记录员找到当前执行到的命令，并将命令提取出来放到命令控制器中的指令暂存处

（2）接着控制器中的命令解释器对命令进行解释，并产生相应的控制信号

（3）在控制器的控制下，将两个数再从存储器中提取出来，分别放到运算器的两个数据缓存区中

（4）接着控制器产生一个控制信号告诉电路做这两个数的加法，相加得到运算结果.

（5）把运算结果写回存储器指定位置。

（6）完这条命令后，转到下一条命令继续执行.

**流水线技术：**

背景：

根据CPU的运行原理我们发现：

取指阶段占用的CPU硬件是系统总线和指令寄存器；

译码阶段占用的CPU硬件是指令译码器；

执行阶段占用的CPU硬件是运算器和系统总线。

三者占用的CPU硬件完全不同，这就形成列三级的流水线：

在对第一条指令译码的时候，可以同时对第二条指令进行取指操作；

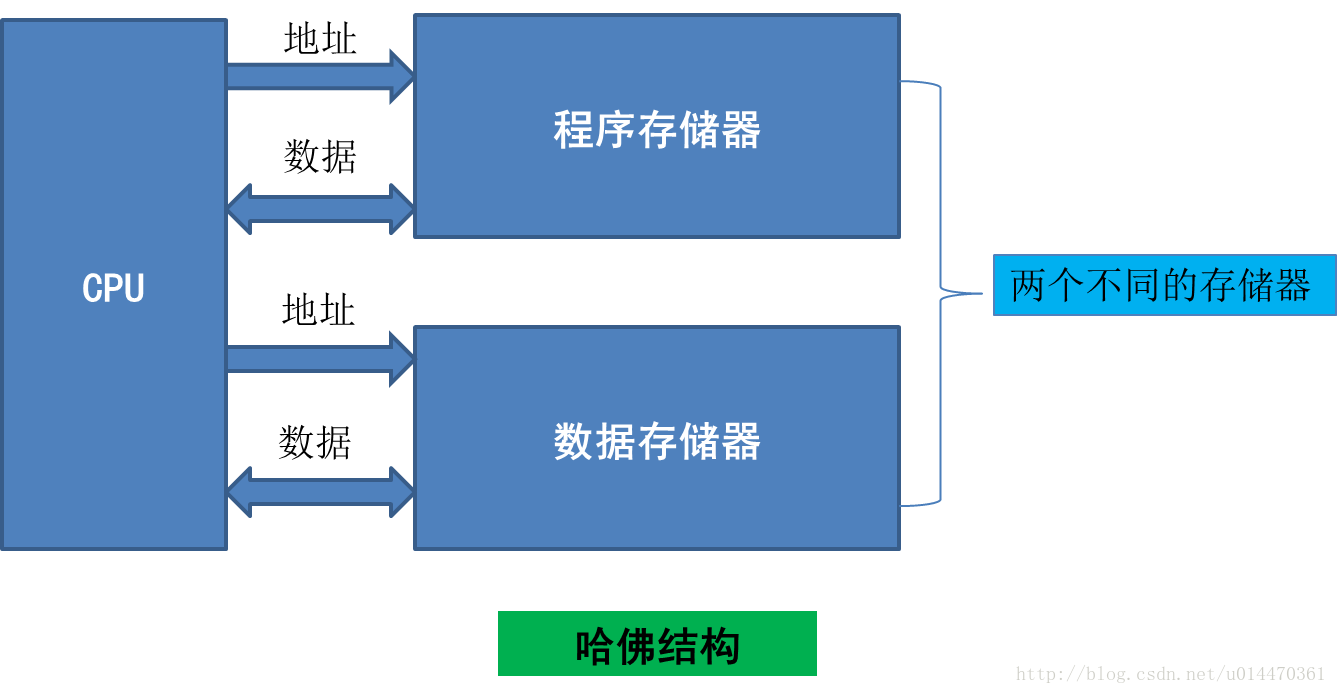
在对第一条指令进行执行的时候，可以同时对第二条进行译码，对第三条进行取指。

概念：

流水线是指在程序执行时多条指令重叠进行操作的一种准并行处理实现技术。

**哈佛结构**

哈佛结构是一种将程序指令存储和数据存储分开的存储器结构，如下图所示。

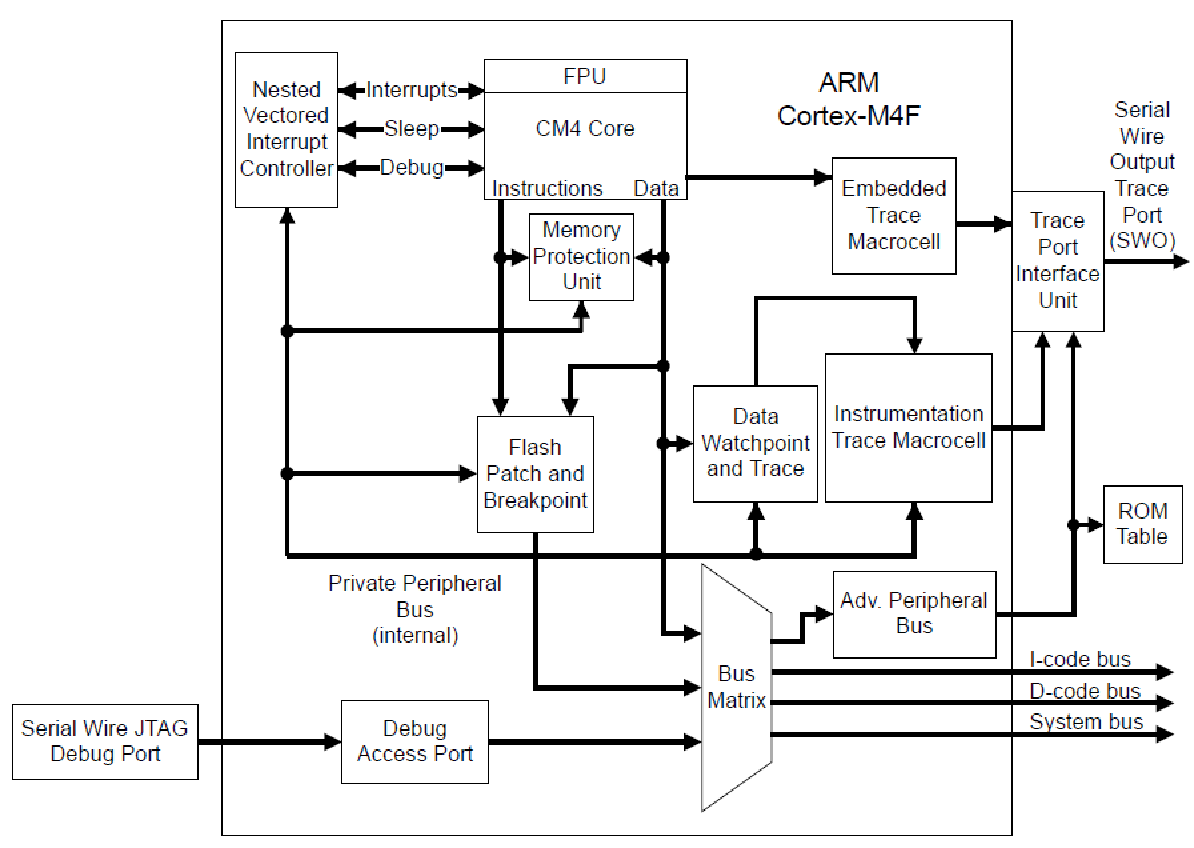


* 中央处理器首先到程序指令存储器中读取程序指令内容，解码后得到数据地址，再到相应的数据存储器中读取数据，并进行下一步的操作（通常是执行）。程序指令存储和数据存储分开，可以使指令和数据有不同的数据宽度。
* 哈佛结构的微处理器通常具有较高的执行效率。其程序指令和数据指令分开组织和存储的，执行时可以预先读取下一条指令。
* 哈佛结构是指程序和数据空间独立的体系结构，目的是为了减轻程序运行时的访存瓶颈。
* 哈佛结构能基本上解决取指和取数的冲突问题。

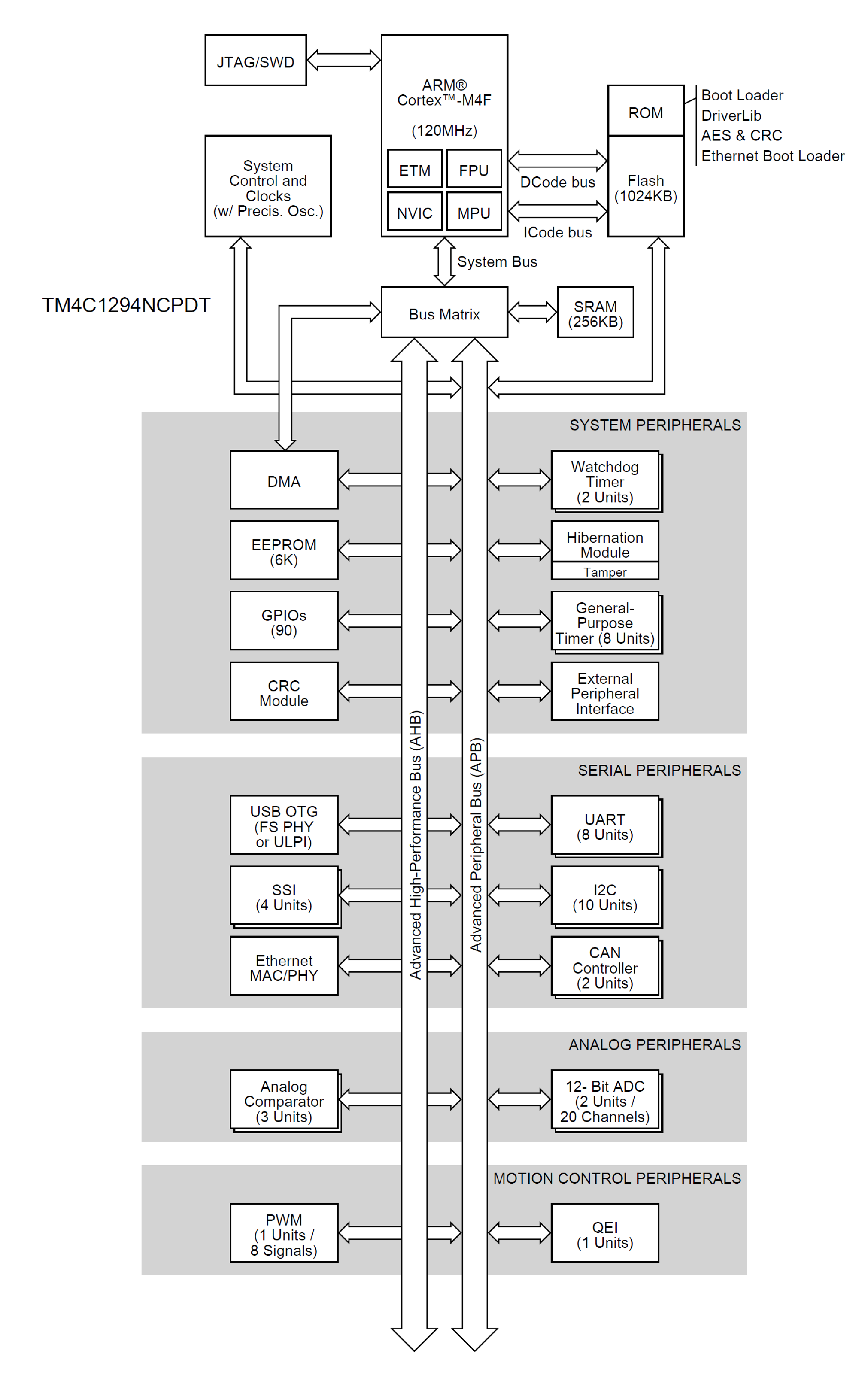
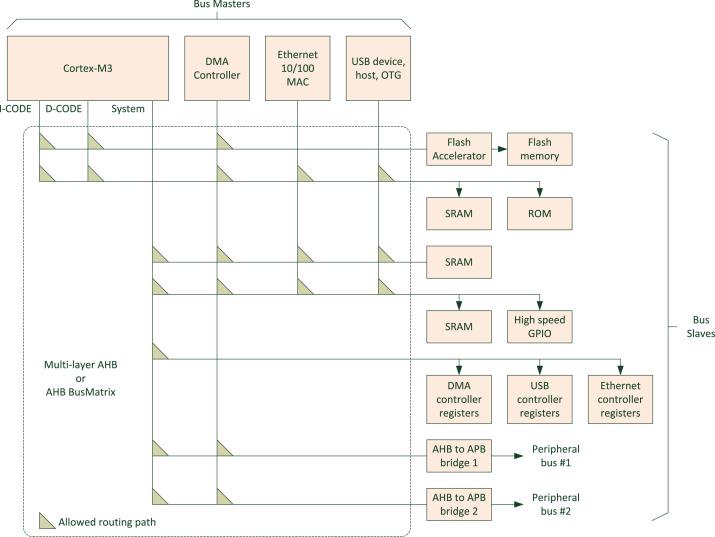
冯·诺依曼体系和哈佛总线体系的区别

* 二者的区别就是程序空间和数据空间是否是一体的。冯·诺依曼结构数据空间和地址空间不分开，哈佛结构数据空间和地址空间是分开的。
* 哈佛总线技术应用是以DSP和ARM为代表的。采用哈佛总线体系结构的芯片内部程序空间和数据空间是分开的，这就允许同时取指和取操作数，从而大大提高了运算能力。

Cortex M4F处理器具有多条总线，是典型的哈佛结构

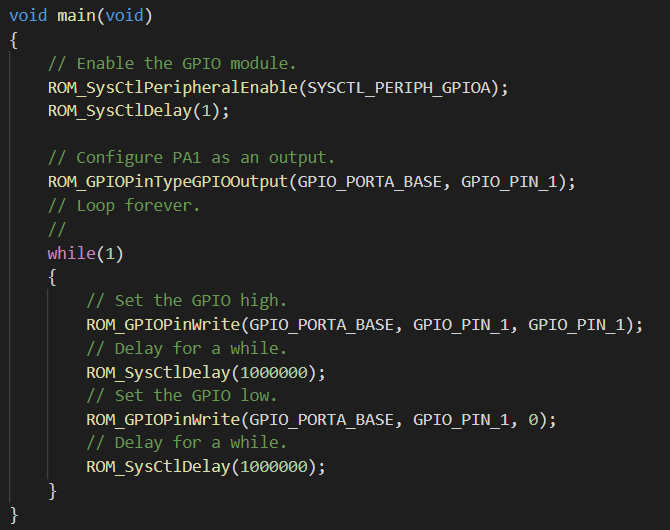


采用总线矩阵的形式提高信息交互的速度



**嵌入式软件C语言开发的特点：**

程序由main函数开始执行，包含一个while无限循环。



**C语言的语法回顾：**

**关键字**：关键字是C语言中具有特殊功能的保留标示符，按照功能可分为

1). 数据类型(常用char, short, int, long, unsigned, float, double)

2). 运算和表达式( =, +, -, \*, while, do-while, if, goto, switch-case)

3). 数据存储(auto， static， extern，const， register，volatile，restricted)，

4). 结构(struct, enum, union,typedef),

5). 位操作和逻辑运算(<<, >>, &, |, ~，^, &&)，

6). 预处理(#define, #include, #error，#if...#elif...#else...#endif等)，

7). 平台扩展关键字(\_\_asm, \_\_inline，\_\_syscall)

**数据类型：**

C语言支持常用的字符型，整型，浮点型变量，C语言只规定了每种基本数据类型的最小取值范围，因此在不同芯片平台上相同类型可能占用不同长度的存储空间，这就需要在代码实现时考虑后续移植的兼容性，而C语言提供的typedef就是用于处理这种情况的关键字，在大部分支持跨平台的软件项目中被采用，典型的如下:

typedef unsigned char uint8\_t;

typedef unsigned short uint16\_t;

typedef unsigned int uint32\_t;

......

typedef signed int int32\_t;

指针的宽度，和芯片的可寻址宽度有关，如32位MCU的宽度就是4，64位MCU的宽度就是8。

**内存管理和存储架构**

C语言允许程序变量在定义时就确定内存地址，通过作用域，以及关键字extern，static，实现了精细的处理机制，按照在硬件的区域不同，内存分配有三种方式(节选自C++高质量编程)：

1). **从静态存储区域分配**。内存在程序编译的时候就已经分配好，这块内存在程序的整个运行期间都存在。例如全局变量，static 变量。

2). **在栈上创建**。在执行函数时，函数内局部变量的存储单元都可以在栈上创建，函数执行结束时这些存储单元自动被释放。栈内存分配运算内置于处理器的指令集中，效率很高，但是分配的内存容量有限。

3). **从堆上分配**，亦称动态内存分配。程序在运行的时候用 malloc 或 new 申请任意多少的内存，程序员自己负责在何时用 free 或 delete 释放内存。动态内存的生存期由程序员决定，使用非常灵活，但同时遇到问题也最多。



C语言的作用域不仅描述了标识符的可访问的区域，其实也规定了变量的存储区域，在文件作用域的变量st\_val和ex\_val被分配到静态存储区，其中static关键字主要限定变量能否被其它文件访问，

而代码块作用域中的变量a, ptr和local\_st\_val则要根据类型的不同，分配到不同的区域，其中a是局部变量，被分配到栈中，ptr作为指针，由malloc分配空间，因此定义在堆中，而local\_st\_val则被关键字限定，表示分配到静态存储区，这里就涉及到重要知识点，static在文件作用域和代码块作用域的意义是不同的：在文件作用域用于限定函数和变量的外部链接性(能否被其它文件访问), 在代码块作用域则用于将变量分配到静态存储区。

对于C语言，如果理解上述知识对于内存管理基本就足够，但对于嵌入式C来说，定义一个变量，它不一定在内存(SRAM)中，也有可能在FLASH空间，或直接由寄存器存储(register定义变量或者高优化等级下的部分局部变量)，如**定义为const的全局变量定义在FLASH中**，**定义为register的局部变量会被优化到直接放在通用寄存器中**，在优化运行速度，或者存储受限时，理解这部分知识对于代码的维护就很有意义。此外，嵌入式C语言的编译器中会扩展内存管理机制，如支持分散加载机制和\_\_attribute\_\_((section("用户定义区域")))，允许指定变量存储在特殊的区域如(SDRAM, SQI FLASH), 这强化了对内存的管理，以适应复杂的应用环境场景和需求。

**指针和数组**

数组是由相同类型元素构成，当它被声明时，编译器就根据内部元素的特性在内存中分配一段空间，另外C语言也提供多维数组，以应对特殊场景的需求，而指针则是提供使用地址的符号方法，只有指向具体的地址才有意义，C语言的指针具有最大的灵活性，在被访问前，可以指向任何地址，这大大方便了对硬件的操作，但同时也对开发者有了更高的要求。参考如下代码：

不同类型指针加法的运算结果:

int main(void)

{

char cval[] = "hello";

int i;

int ival[] = {1, 2, 3, 4};

int arr\_val[][2] = {{1, 2}, {3, 4}};

const char \*pconst = "hello";

char \*p;

int \*pi;

int \*pa;

int \*\*par;

p = cval;

p++; //addr增加1

pi = ival;

pi+=1; //addr增加4

pa = arr\_val[0];

pa+=1; //addr增加4

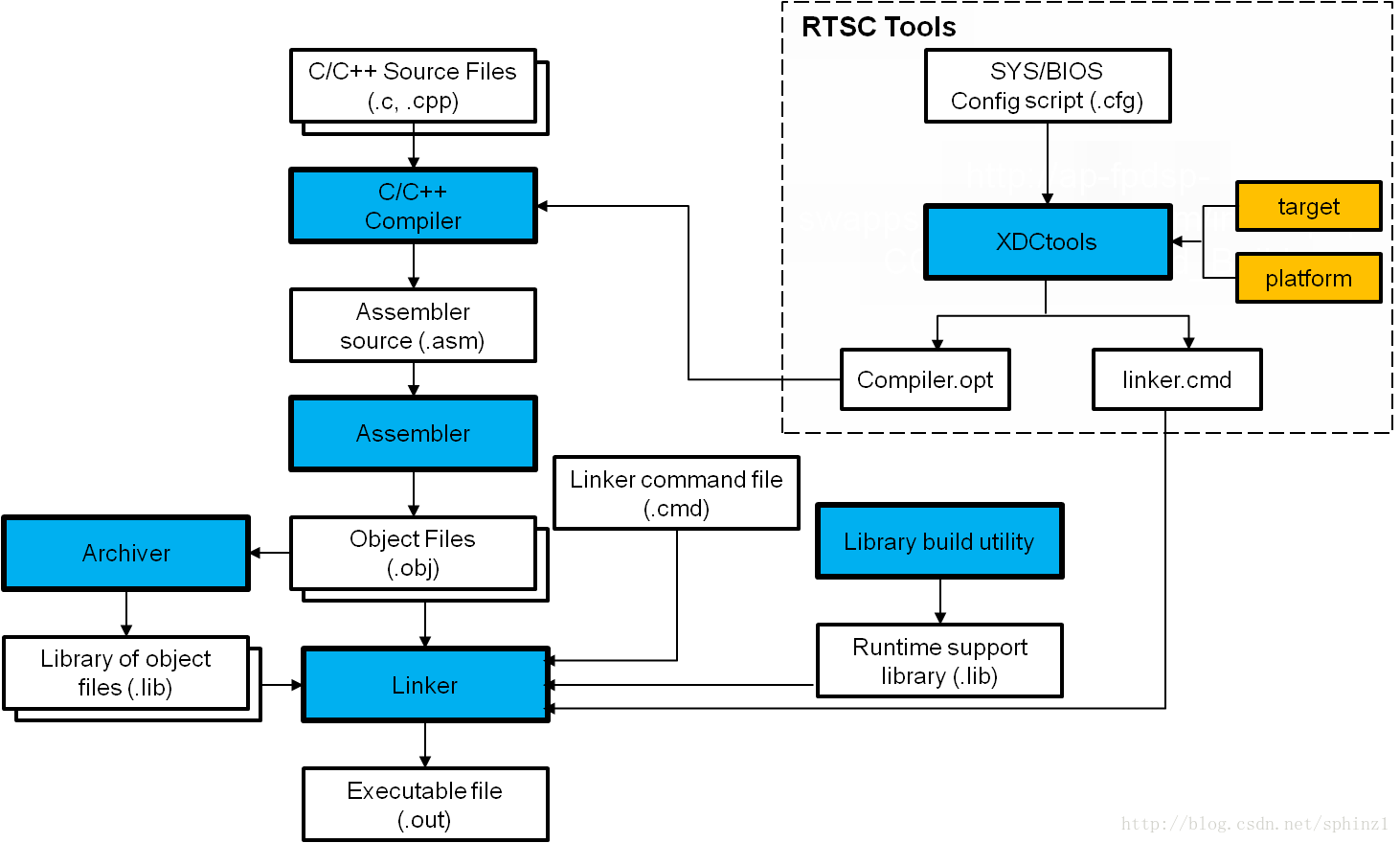
par = arr\_val;

par++; //addr增加8

}

**C语言的编译过程：**

* 预处理(Preprocessing)，即完成宏定义和include文件展开等工作；
* 编译(Compilation)，根据编译参数进行不同程序的优化，编译成汇编代码；
* 汇编(Assemble)，用汇编器把上一阶段生成的汇编代码进一步生成目标代码；
* 链接(Linking)，用链接器把上一阶段生成的目标代码、其他一些相关的系统提供的目标代码（如crtx.o）和系统或用户提供的库链接起来，生成最终的执行代码。生成可执行文件



编译过程生成的文件：

